

CSE211 Digital Design

Akdeniz University

Week2-3: Introduction to Digital Design

1

Assoc.Prof.Dr. Taner Danişman

tdanisman@akdeniz.edu.tr

Course program

2

Week 01	19-Sep-22	Introduction
Week 02	26-Sep-22	Digital Systems and Binary Numbers I
Week 03	03-Oct-22	Digital Systems and Binary Numbers II
Week 04	10-Oct-22	Boolean Algebra and Logic Gates I
Week 05	17-Oct-22	Boolean Algebra and Logic Gates II
Week 06	24-Oct-22	Boolean Algebra and Logic Gates III
Week 07	31-Oct-22	Gate Level Minimization
Week 08	07-Nov-22	Midterm
Week 09	14-Nov-22	Karnaugh Maps
Week 10	21-Nov-22	Karnaugh Maps
Week 11	28-Nov-22	Combinational Logic
Week 12	05-Dec-22	Combinational Logic
Week 13	12-Dec-22	Timing, delays and hazards
Week 14	19-Dec-22	Synchronous Sequential Logic
Week 15	26-Dec-22	Arduino Programming

Number Systems

➤ Decimal Numbers

➤ What does 5,634 represent?

➤ Expanding 5,634:

$$5 \times 10^3 = 5,000$$

$$+ 6 \times 10^2 = 600$$

$$+ 3 \times 10^1 = 30$$

$$+ 4 \times 10^0 = 4 \rightarrow 5,634$$

➤ What is “10” called in the above expansion?

➤ The radix.

➤ What is this type of number system called?

➤ Decimal.

➤ What are the digits for decimal numbers?

➤ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

➤ What are the digits for radix-r numbers?

➤ 0, 1, ..., r-1.

Example: Convert 46.6875_{10} To Base 2

► Convert 46 to Base 2

$$46/2 = 23 \text{ remainder} = 0$$

$$23/2 = 11 \text{ remainder} = 1$$

$$11/2 = 5 \text{ remainder} = 1$$

$$5/2 = 2 \text{ remainder} = 1$$

$$2/2 = 1 \text{ remainder} = 0$$

$$1/2 = 0 \text{ remainder} = 1$$

Read off in reverse order: 101110_2

► Convert 0.6875 to Base 2:

$$0.6875 * 2 = 1.3750 \text{ int} = 1$$

$$0.3750 * 2 = 0.7500 \text{ int} = 0$$

$$0.7500 * 2 = 1.5000 \text{ int} = 1$$

$$0.5000 * 2 = 1.0000 \text{ int} = 1$$

$$0.0000$$

Read off in forward order: 0.1011_2

► Join together with the radix point: 1011110.1011_2

Converting Among Octal, Hexadecimal, Binary

➤ Octal (Hexadecimal) to Binary:

- Restate the octal (hexadecimal) as three (four) binary digits, starting at radix point and going both ways

➤ Binary to Octal (Hexadecimal):

- Group the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part
- Convert each group of three (four) bits to an octal (hexadecimal) digit

➤ Example: Octal to Binary to Hexadecimal

6 3 5 . 1 7 7 ₈

= 110 | 011 | 101 . 001 | 111 | 111 ₂

= 1 | 1001 | 1101 . 0011 | 1111 | 1(000) ₂ (regrouping)

= 1 9 D . 3 F 8 ₁₆ (converting)

BCD

6

- The code is also known as 8-4-2-1 code.
- This is because 8, 4, 2, and 1 are the weights of the four bits of the BCD code
- Since four binary bits are used the maximum decimal equivalent that may be coded is 15_{10} (i.e., 1111_2).
- But the maximum decimal digit available is 9_{10}
- Hence the binary codes 1010, 1011, 1100, 1101, 1110, 1111, representing 10, 11, 12, 13, 14, and 15 in decimal are never being used in BCD code
- These six codes are called **forbidden codes**

BCD

7

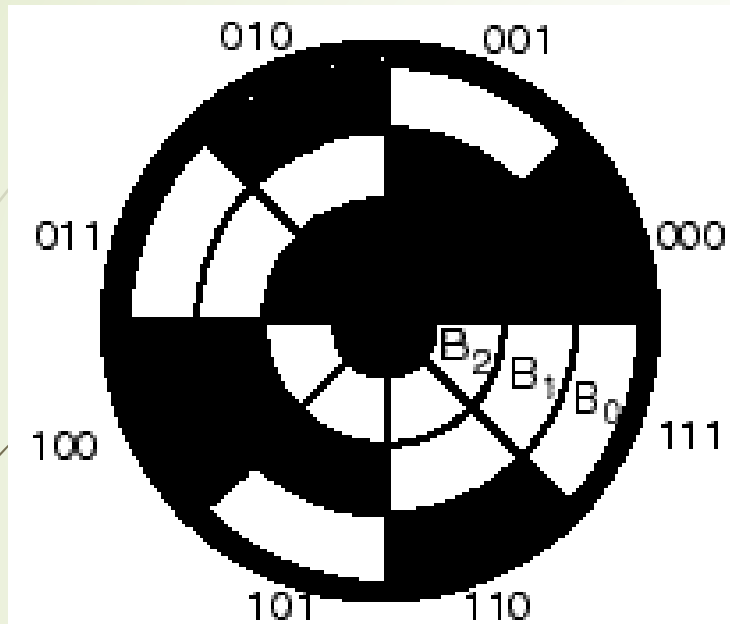
- **Example 2.1** Give the BCD equivalent for the decimal number 589.
- **Solution.**
 - The decimal number is 589 BCD code is 0101 1000 1001 Hence,
 $(589)_{10} = (010110001001)_{\text{BCD}}$

Gray Code

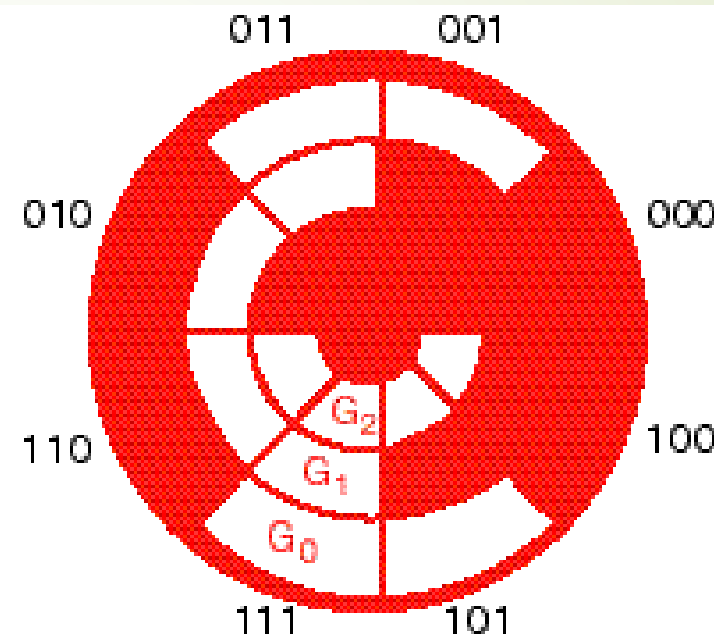
Decimal	8,4,2,1	Gray
0	0000	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

- **What property does this Gray code have?**
 - Counting up or down changes only one bit at a time (including counting between 9 and 0)

Gray Code: Optical Shaft Encoder



(a) Binary Code for Positions 0 through 7



(b) Gray Code for Positions 0 through 7

- Shaft encoder: Capture angular position (e.g., compass)
- For binary code, what values can be read if the shaft position is at boundary of "3" and "4" (011 and 100) ?
- For Gray code, what values can be read ?

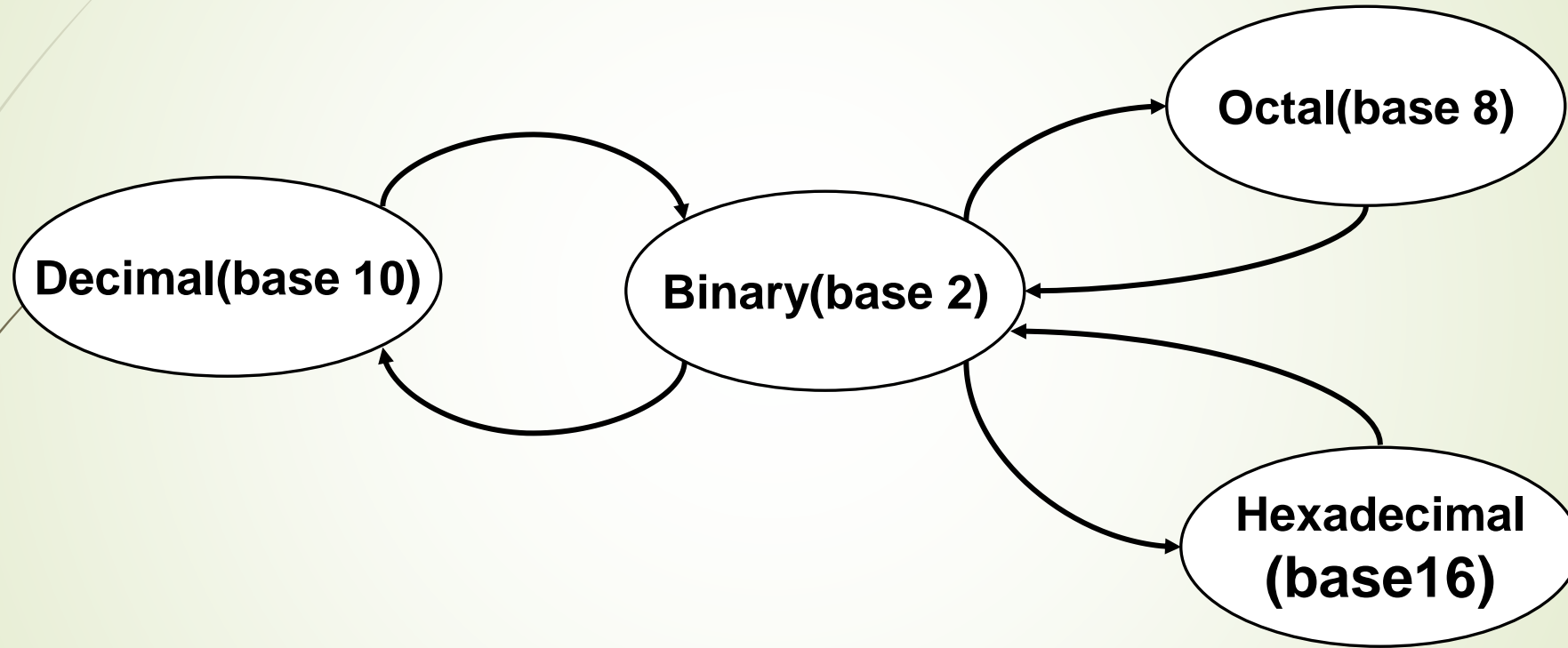
Warning: Conversion or Coding?

10

- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$ (This is conversion)
- $13 \Leftrightarrow 0001 \mid 0011$ (This is coding)

Conversion Between Number Bases

11



Conversion of a Binary Number into Gray Code

- the MSB of the Gray code is the same as the MSB of the binary number;
- the second bit next to the MSB of the Gray code equals the Ex-OR of the MSB and second bit of the binary number; it will be 0 if there are same binary bits or it will be 1 for different binary bits;
- the third bit for Gray code equals the exclusive-OR of the second and third bits of the binary number, and similarly all the next lower order bits follow the same mechanism

Conversion of a Binary Number into Gray Code

➤ **Example 2.2** Convert $(101011)_2$ into Gray code.

Solution.

Step 1. The MSB of the Gray code is the same as the MSB of the binary number.

1	0	1	0	1	1	Binary
↓						
1						Gray

Step 2. Perform the ex-OR between the MSB and the second bit of the binary. The result is 1, which is the second bit of the Gray code.

1	⊕	0	1	0	1	1	Binary
↓							
1	1						Gray

Step 3. Perform the ex-OR between the second and the third bits of the binary. The result is 1, which is the third bit of the Gray code.

1		0	⊕	1	0	1	1	Binary
↓								
1	1	1						Gray

Step 4. Perform the ex-OR between the third and the fourth bits of the binary. The result is 1, which is the fourth bit of the Gray code.

1	0	1	⊕	0	1	1	Binary
↓							
1	1	1	1				Gray

Step 5. Perform the ex-OR between the fourth and the fifth bits of the binary. The result is 1, which is the fifth bit of the Gray code.

1	0	1	0	⊕	1	1	Binary
↓							
1	1	1	1	1			Gray

Step 6. Perform the ex-OR between the fifth and the sixth bits of the binary. The result is 0, which is the last bit of the Gray code.

1	0	1	0	1	⊕	1	Binary
↓							
1	1	1	1	1	0		Gray

Conversion of a Binary Number into Gray Code

- Convert $(564)_{10}$ into Gray code.
- Decimal number 564 is equal to the Binary number



Some chapter terms to know

15

- DVD
- GUI
- BCD
- ASCII
- STX
- ETX
- CR
- HDL

Binary Arithmetic

16

- Single Bit Addition with Carry
- Multiple Bit Addition
- Single Bit Subtraction with Borrow
- Multiple Bit Subtraction
- Multiplication
- BCD Addition

Binary Addition

- Binary addition is very simple.
- This is best shown in an example of adding two binary numbers...

$$\begin{array}{r} 111111 \leftarrow \text{carries} \\ 111101 \\ + 10111 \\ \hline 1010100 \end{array}$$

Binary addition

18

➤ Same as decimal addition

➤ $0+0=0$

➤ $0+1=1$

➤ $1+0=1$

➤ $1+1=0+C$

$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$	$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$	$\begin{array}{r} 1 \\ + 1 \\ \hline 11 \end{array}$	$\begin{array}{r} 1 \\ + 10 \\ \hline 11 \end{array}$	$\begin{array}{r} 10 \\ + 101 \\ \hline 111 \end{array}$	$\begin{array}{r} 1011 \\ + 10 \\ \hline 1101 \end{array}$	$\begin{array}{r} 101011 \\ + 111 \\ \hline 110010 \end{array}$	$\begin{array}{r} \textcircled{1}\textcircled{1}\textcircled{1}\textcircled{1} \\ 101011 \\ + 111 \\ \hline 110010 \end{array}$
---	---	---	--	--	---	--	--	---	---

19

- We can also perform subtraction (with borrows in place of carries).
- Let's subtract $(10111)_2$ from $(1001101)_2$...

$$\begin{array}{r}
 1 \\
 0 \cancel{10} 10 0 \cancel{0} 10 \leftarrow \text{borrows} \\
 \cancel{1} \cancel{0} \cancel{0} \cancel{1} \cancel{1} \cancel{0} 1 \\
 - 1 0 1 1 1 \\
 \hline
 1 1 0 1 1 0
 \end{array}$$

Binary Subtraction

20

➤ $0-0=0$

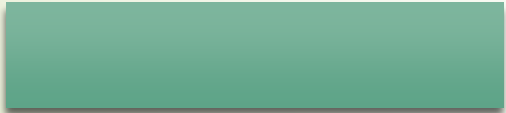
➤ $1-0=1$

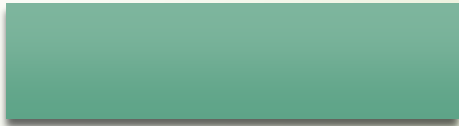
➤ $1-1=0$

➤ $0-1=0+B$

➤ B(Borrow)

$\begin{array}{r} 0 \\ - 0 \\ \hline 0 \end{array}$	$\begin{array}{r} 1 \\ - 0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ - 1 \\ \hline 0 \end{array}$	$\begin{array}{r} 10 \\ - 1 \\ \hline 01 \end{array}$	$\begin{array}{r} 101 \\ - 10 \\ \hline 011 \end{array}$
---	---	---	---	--

$$\begin{array}{r} 11001010 \\ - 10011011 \\ \hline \end{array}$$


$$\begin{array}{r} 1001 \\ - 10 \\ \hline \end{array}$$


Binary Multiplication

- Binary multiplication is much the same as decimal multiplication, except that the multiplication operations are much simpler...

$$\begin{array}{r} 1 0 1 1 1 \\ X 1 0 1 0 \\ \hline 0 0 0 0 0 \\ 1 0 1 1 1 \\ 0 0 0 0 0 \\ 1 0 1 1 1 \\ \hline 1 1 1 0 0 1 1 0 \end{array}$$

1's and 2's Complements

- Complements are used in digital computers to simplify the subtraction operation and for logical manipulation.
- Simplifying operations leads to simpler, less expensive circuits to implement the operations
- There are two types of complements for each **base-r** system:
 - the **diminished radix complement (r-1's complement)**
 - the **radix complement (r's complement)**

One's Complement Representation

- The one's complement of a binary number involves inverting all bits.
 - 1's comp of 00110011 is 11001100
 - 1's comp of 10101010 is 01010101
- Called **diminished radix complement** by Mano
- For an n bit number N the 1's complement is $(2^n - 1) - N$
 - e.g. For decimal numbers $r=10$ and $r-1=9$ So **9's** complement of N is $(10^n - 1) - N$
 - In this case, 10^n represents a number that consists of a single 1 followed by n 0's
 - $10^n - 1$ is a number represented by n 9's
 - if $n = 4$, we have $10^4 = 10,000$ and $10^4 - 1 = 9999$
- To find negative of 1's complement number take the 1's complement.

$00001100_2 = 12_{10}$
 ↑ ↑
 Sign bit Magnitude

$11110011_2 = -12_{10}$
 ↑ ↑
 Sign bit Magnitude

Two's Complement Representation

24

- The two's complement of a binary number involves inverting all bits and adding 1.
 - 2's comp of 00110011 is 11001101
 - 2's comp of 10101010 is 01010110
- Called radix complement by Mano
- For an n bit number N the 2's complement is $(2^n - 1) - N + 1$
 - the 10's complement of decimal **2389** is **7610 + 1 = 7611** and is obtained by adding 1 to the 9's complement value
 - The 2's complement of binary **101100** is **010011 + 1 = 010100** and is obtained by adding 1 to the 1's-complement value
- To find negative of 2's complement number take the 2's complement.

$$\begin{array}{c} \text{Sign bit} \nearrow \quad \nwarrow \text{Magnitude} \\ \underline{00001100}_2 = 12_{10} \end{array}$$

$$\begin{array}{c} \text{Sign bit} \nearrow \quad \nwarrow \text{Magnitude} \\ \underline{11110100}_2 = -12_{10} \end{array}$$

Two's Complement Shortcuts

25

- Algorithm 1 – Simply complement each bit and then add 1 to the result.
Finding the 2's complement of $(01100101)_2$ and of its 2's complement...

N	=	01100101		[N]	=	10011011		
		10011010				01100100		
		+		1		+		1
		-----				-----		
		10011011				01100101		

- Algorithm 2 – Starting with the least significant bit, copy all of the bits up to and including the first 1 bit and then complementing the remaining bits.

N	=	0 1 1 0 0 1 0 1
[N]	=	1 0 0 1 1 0 1 1

1's Complement Addition

26

- Using 1's complement numbers, adding numbers is easy.
- For example, suppose we wish to add +12 and +1
- Let's compute $(12)_{10} + (1)_{10}$

$$(12)_{10} = +(1100)_2 = 01100_2 \text{ in 1's comp.}$$

$$(1)_{10} = +(0001)_2 = 00001_2 \text{ in 1's comp.}$$

Step 1: Add binary numbers
Step 2: Add carry to low-order bit

				0	1	1
				0	0	0
				0	0	1

				0	0	1
				1	1	0
				1	0	1

				0	1	1
				0	1	0
				0	1	1

Add carry

Final Result

1's Complement Subtraction

27

- Using 1's complement numbers, subtracting numbers is also easy.
- For example, suppose we wish to subtract +1 from +12
- Let's compute $(12)_{10} - (1)_{10}$.

$$(12)_{10} = +(1100)_2 = 01100_2 \text{ in 1's comp.}$$

$$(-1)_{10} = -(0001)_2 = 11110_2 \text{ in 1's comp.}$$

Step 1: Take 1's complement of 2nd operand

Step 2: Add binary numbers

Step 3: Add carry to low order bit

Diagram illustrating the 1's complement subtraction process:

-	0	1	1	0	0
	0	0	0	0	1

+	0	1	1	0	0
	1	1	1	1	0

	1	0	1	0	0
					1

	0	1	0	1	1

Final Result: 0 1 0 1 1

2's Complement Addition

28

- Using 2's complement numbers, adding numbers is easy.
- For example, suppose we wish to add +12 and +1
- Let's compute $(12)_{10} + (1)_{10}$

$$(12)_{10} = +(1100)_2 = 01100_2 \text{ in 2's comp.}$$

$$(1)_{10} = +(0001)_2 = 00001_2 \text{ in 2's comp.}$$

Step 1: Add binary numbers
Step 2: Ignore carry bit

						0 1 1 0 0
						0 0 0 0 1

						0 0 1 1 0 1
					↑	
					Ignore	

Final Result

2's Complement Subtraction

- Using 2's complement numbers, follow steps for subtraction
- For example, suppose we wish to subtract +1 from +12
- Let's compute $(12)_{10} - (1)_{10}$

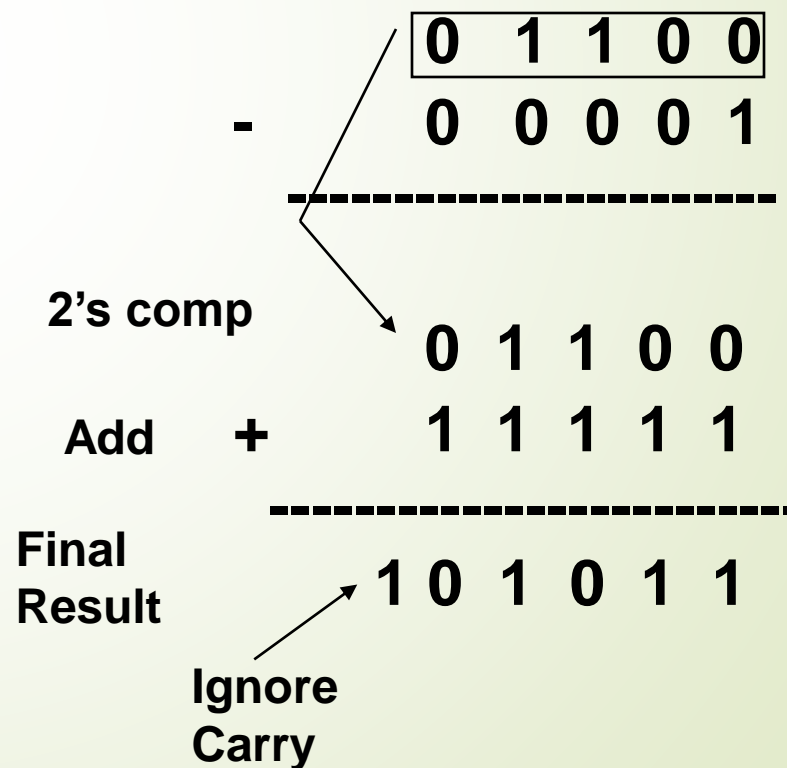
$$(12)_{10} = +(1100)_2 = 01100_2 \text{ in 2's comp.}$$

$$(-1)_{10} = -(0001)_2 = 11111_2 \text{ in 2's comp.}$$

Step 1: Take 2's complement of 2nd operand

Step 2: Add binary numbers

Step 3: Ignore carry bit



2's Complement Subtraction: Example # 2

- Let's compute $(13)_{10} - (5)_{10}$.

$$(13)_{10} = +(1101)_2 = (01101)_2$$

$$(-5)_{10} = -(0101)_2 = (11011)_2$$

- Adding these two 5-bit codes...

carry

+

0 1 1 0 1
1 1 0 1 1

1 0 1 0 0 0

- Discarding the carry bit, the sign bit is seen to be zero, indicating a correct result. Indeed,

$$(01000)_2 = +(1000)_2 = +(8)_{10}.$$

2's Complement Subtraction:

Example #3

- Let's compute $(5)_{10} - (12)_{10}$.
 $(-12)_{10} = -(1100)_2 = (10100)_2$
 $(5)_{10} = +(0101)_2 = (00101)_2$
- Adding these two 5-bit codes...

$$\begin{array}{r} \\ + \\ \hline 1 \end{array}$$

- Here, there is **no carry bit** and the sign bit is 1. This indicates a negative result, which is what we expect. $(11001)_2 = -(7)_{10}$.

2's Complement Subtraction:

32

Example #4

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction **(a)** $X - Y$ and **(b)** $Y - X$ by using 2's complements.

$$(a) \quad X = 1010100$$

$$2\text{'s complement of } Y = + \underline{0111101}$$

$$\text{Sum} = 10010001$$

$$\text{Discard end carry } 2^7 = - \underline{10000000}$$

$$\text{Answer: } X - Y = 0010001$$

$$(b) \quad Y = 1000011$$

$$2\text{'s complement of } X = + 0101100$$

$$\text{Sum} = 1101111$$

There is no end carry. Therefore, the answer is $Y - X = -(2\text{'s complement of } 1101111) = -0010001$.

10's Complement Subtraction: Example #5

- Let's compute $(72532)_{10} - (3250)_{10}$.

$$99999 - 3250$$

$$96749 + 1 = 96750$$

Using 10's complement, subtract $72532 - 3250$.

$$M = 72532$$

$$10\text{'s complement of } N = + \underline{96750}$$

$$\text{Sum} = 169282$$

$$\text{Discard end carry } 10^5 = - \underline{100000}$$

$$\text{Answer} = 69282$$

10's Complement Subtraction:

34

Example #6

- Let's compute $(3250)_{10} - (72532)_{10}$.

$$99999 - 72532$$

$$27467 + 1 = 27468$$

Using 10's complement, subtract $3250 - 72532$.

$$M = 03250$$

$$10\text{'s complement of } N = + \underline{27468}$$

$$\text{Sum} = 30718$$

There is no end carry. Therefore, the answer is $-(10\text{'s complement of } 30718) = -69282$

BCD Addition

- ▶ When the binary sum is equal to or less than 1001 (without a carry), the corresponding BCD digit is correct
- ▶ However, when the binary sum is greater than or equal to 1010, the result is an invalid BCD digit
- ▶ The addition of $6 = (0110)_2$ to the binary sum converts it to the correct digit and also produces a carry as required

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	1001
<u>9</u>	<u>1001</u>	<u>12</u>	<u>1100</u>	<u>17</u>	<u>10001</u>
			+0110		+0110
			<u>10010</u>		<u>10111</u>

BCD Addition

- If the binary sum is greater than or equal to 1010, we add 0110 to obtain the correct BCD sum and a carry

Consider the addition of $184 + 576 = 760$ in BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	<u>0111</u>	<u>0110</u>	+576
Binary sum	0111	10000	1010	
Add 6		<u>0110</u>	<u>0110</u>	
BCD sum	0111	0110	0000	<u>760</u>

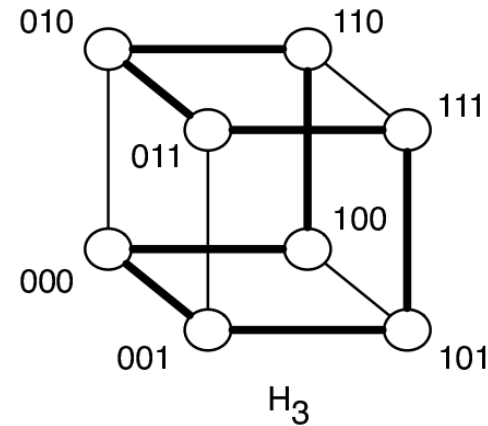
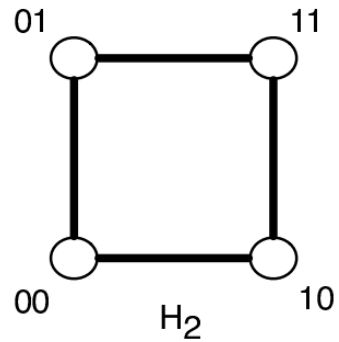
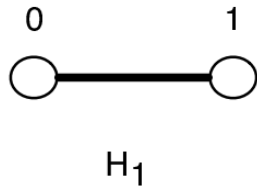
BCD Subtraction

37

- We use 10's complement for the BCD subtraction. Positive numbers represented by **0000** while negative numbers represented by **1001**
- **357-432=?**
- **357** in BCD is **0000 0011 0101 0111**
- **-432** in **10's** complement is **999 - 432 = 567**, and **567 + 1 = 568**
- Since this value is negative we add trailing **1001** to **0101 0110 1000**
- | | | | | | | | | | | | | | |
|-------------|-------------|-------------|-------------|----------|-------------|-------------|-------------|-------------|----------|-------------|-------------|-------------|-------------|
| 0000 | 0011 | 0101 | 0111 | + | 1001 | 0101 | 0110 | 1000 | = | 1001 | 1000 | 1011 | 1111 |
| 0 | 3 | 5 | 7 | + | 9 | 5 | 6 | 8 | = | 9 | 8 | 11 | 15 |
- We add 6 to the nibble values greater than **1001**.
- The we obtained **1001 1001 0010 0101** which is **-925**.
- In order to find the magnitude: **999 - 925 = 74**, and **74 + 1 = 75**
- Then final result is **357 - 432 = -75**

Questions

- What coding type might be used in the following figures?



- How can we find 16th gray code value?

Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15